

APPENDIX A

```

/*
 * (C) Copyright 2000 by OnMercial.com, Inc.
5  * All rights reserved
 *
 * General pixel encoding routine.
 *
 * The routine ac_out is a standard range encoder. The routine UpdateModel
10 * is responsible for updating the model to reflect the new probability
 * values given the color of the current pixel.
 */
int AmfCoder::encode(int a_y,int a_x) {
/*
15 * pc is the index into the color palette for the current pixel.
 * pl is the index into the color palette for the left neighbor pixel.
 * plu is the index into the color palette for the upper neighbor pixel.
 * ptr is a two-dimensional array storing the indices into the color
 * palette in the image.
20 */
    int pc;
    int pl;
    int pu;
    int ptr;
25 /*
 * Identify the index into the color palette for the current pixel.
 */
    pc=(*frame)(a_x,a_y);
/*
30 * Assert the proposition that the index into the color palette for the
 * current pixel is a valid index (i.e., not out of bounds).
 */
    assert(pc<pal_size+1);
/*
35 * Identify the indices into the color palette for the upper neighbor.
 */
    if(a_y>frame->y0) {
        pu=(*frame)(a_x,a_y-1);
    } else {
40 /*
 * Use the index in the color palette for the transparent color to indicate
 * that the current pixel is in the top row of the image.
 */
        pu=pal_size;
45 }
/*
 * Identify the indices into the color palette for the left neighbor.
 */
    if(a_x>frame->x0) {
50         pl=(*frame)(a_x-1,a_y);
    } else {
/*
 * Use the index in the color palette for the transparent color to indicate
 * that the current pixel is in the top row of the image.
55 */
        pl=pal_size;
    }
/*
 * Encode the current pixel.
60 */

```

```

* Mode 1: The left and upper pixels have the same color.
*/
  if (pu==pl) {
/*
5  * Determine the proper offsets into the frq_eq array for the current
  * probability model.
  */
    ptr=3*context;
/*
10 * Mode 1a: The current pixel has the same color as the left and upper
   * neighbors.
   */
    if (pl==pc) {
/*
15 * Encode the mode and update the model.
    */
      coder->ac_out(frq_eq[ptr+0],frq_eq[ptr+1],frq_eq[ptr+2]);
      UpdateModel(ptr+frq_eq,2,0,ALEVEL);
/*
20 * Select the proper model for the next pixel.
    */
      context=0;
/*
25 * Mode 1b: The current pixel has a different color than the left and upper
   * neighbors.
   */
    } else {
/*
30 * Encode the mode and the index into the color palette for the current
   * pixel, and update the model.
   */
      coder->ac_out(frq_eq[ptr+1],frq_eq[ptr+2],frq_eq[ptr+2]);
      encode_0(pl,pl,pc);
      UpdateModel(ptr+frq_eq,2,1,ALEVEL);
35 /*
   * Select the proper model for the next pixel.
   */
      context=1;
    }
40 /*
   * Mode 2: The left and upper pixels have different colors.
   */
  } else {
/*
45 * Determine the proper offsets into the frq_ne array for the current
   * probability model.
   */
    ptr=4*context;
/*
50 * Mode 2a: The current pixel has the same color as the left neighbor only.
   */
    if (pl == pc) {
/*
   * Encode the mode and update the model.
55 */
      coder->ac_out(frq_ne[ptr+0],frq_ne[ptr+1],frq_ne[ptr+3]);
      UpdateModel(ptr+frq_ne,3,0,ALEVEL2);
/*
   * Select the proper model for the next pixel.

```

```

    */
    context=2;
/*
5  * Mode 2b: The current pixel has the same color as the upper neighbor
  * only.
  */
    } else if (pu == pc) {
/*
10  * Encode the mode and update the model.
  */
    coder->ac_out(frq_ne[ptr+1], frq_ne[ptr+2], frq_ne[ptr+3]);
    UpdateModel(ptr+frq_ne, 3, 1, ALEVEL2);
/*
15  * Select the proper model for the next pixel.
  */
    context=3;
/*
20  * Mode 2c: The current pixel has a different color than the left and upper
  * neighbors.
  */
    } else {
/*
25  * Encode the mode and the index into the color palette for the current
  * pixel, and update the model.
  */
    coder->ac_out(frq_ne[ptr+2], frq_ne[ptr+3], frq_ne[ptr+3]);
    UpdateModel(ptr+frq_ne, 3, 2, ALEVEL2);
    encode_0(pl, pu, pc);
30  */
    * Select the proper model for the next pixel.
  */
    context=4;
    }
35  }
    return 0;
}

40
/*
  * Encoding routine for encoding the current pixel when the current pixel
  * has a color different from both the left and upper neighbors.
  */
45  int AmfCoder::encode_0(int a_l, int a_u, int a_c) {
/*
  * xl marks the low end of the range of probability values for a color
  *   in the color palette.
  *   xh marks the high end of the range of probability values for a color
50  *   in the color palette.
  *   xtota stores the total number of occurrences of each color in the
  *   color palette.
  */
    U16B xl;
55    U16B xh;
    U16B xtota;

    U16B i;
/*
60  * Initialize the low end of the range of probability values for the first
  * color in the color palette.

```

```

    */
    xl=0;
/*
5  * Calculate the total number of occurrences of each color in the color
    * palette.
    */
    xtot=frq_0[pal_size+1];
/*
10 * Exclude from the total number of occurrences of each color in the color
    * palette the number of occurrences of the colors of the left and upper
    * neighbors.
    */
    xtot -= frq_0[a_l];
/*
15 * Only exclude the number of occurrences of the color of the upper
    * neighbor if it is different from the color of the left neighbor (i.e.,
    * we are not in mode 1).
    */
    if (a_l != a_u) {
20         xtot -= frq_0[a_u];
    }
/*
    * Scan through the colors in the color palette.
    */
25 for (i=0;i<pal_size+1;i++) {
/*
    * Ignore the colors or the left and upper neighbors.
    */
    if (i==a_l || i==a_u) {
30         continue;
    }
/*
    * Set the high end of the range of probability values for the current
    * color to be the low end of the range of probability values for the
    * current color plus the number of occurrences of the current color.
35 */
    xh=xl+frq_0[i];
/*
    * If the current color is that of the current pixel, encode the current
    * color and update the model (including the number of occurrences of each
    * color in the color palette).
    */
    if (i==a_c) {
45         coder->ac_out(xl, xh, xtot);
        UpdateModel2(frq_0, (U16B) (pal_size+1), i, ALEVEL0);
    }
/*
    * Break out of the for-loop.
    */
    break;
50 }
/*
    * Set the low end of the range of probability values for the next color to
    * be the high end of the range of probability values for the current
    * color.
55 */
    xl=xh;
}
return 0;
}

```